2016

Mushroom Hunting Application

FINAL PROJECT

PETTERI PESONEN, KRISTIN KARLSSON, HITOMI INABA, RAUL CLARK

SCHOOL OF ARCHITECTURE AND BUILT ENVIRONMENT

Table of contents

1. Introduction

Mushroom picking has become more than just a hobby in Sweden. The number of aficionados grows every year with the advent of health as a lifestyle, and the shifting from a daily activity linked to food supply to a mix between a therapeutic activity (for relaxation purposes) and a widespread collective experience to share with family and friends.

With the wide spreading of geographical information based technologies for personal utilization new possibilities has opened for the activity. Smartphones with apps like google maps and other similar apps have converted mushrooming from personal and family/friends related to a social activity to share with people with similar interests. Apps like Naturplats (Sundfors, 2009) help users to store their findings and favorite places, but also to share it with other people. These apps also help with navigation in natural areas and saving not only detailed information of mushroom findings but also pictures and coordinates of places of interest. This is the main idea behind our Mushrooming application, to build an app that help the user with navigation to pre-existent mushroom picking places, store new findings for further use, analyze these findings and to share it with other users.

2. Application structure

2.1 Choice of Design

The basic idea of the application design (Figure 5) is to be as simple as possible without any unnecessary clutter such as headers and footers. Therefore, we ended up only using a simple sidebar that functions as a container for all of our functionalities. The main view (Figure 7) is composed of different elements, the map where the data is displayed and the sidebar where the functionalities are accessed. These functionalities are directed into separate pages (Figure 8) to provide helpful information to users that might not be acquainted with the application, but also to divide the functionalities according to their similarities:

- Closest mushroom
- Mushroom findings (different available tools)
- Space-Time Analysis (different available tools)
- Settings
- Sign out

These five groups visually comprise all of the functionalities the user have access to. The first two are related to finding mushroom sites and their access, also to store the places where one discover new picking places so other users have access to them. That way the user gets an ever changing where there is up to date information available at all times, and where the users are a part not only of the experience of enjoying a hobby or an outdoor collective activity (so much in trend in Sweden nowadays), but to become a part of the process of maintaining the application.

The Space-Time Analysis as the name states, is a tool that provides spatial analysis capabilities useful to the users. First, the user can draw polygons to see the statistics of certain areas. It is also possible to create heatmaps of mushroom findings to see where are the more probabilities of finding a determine specie. And last, it is also possible to get animations of the behavior of mushroom sightseeing over time. Space-time analysis tools give an idea of the spatial behavior of the mushroom species, and indirectly of the user preferences and behavior toward mushroom picking.

So far all the applications deal with mushroom related functionalities. Setting and sign out tabs are functionalities oriented to further enhance the completion of the application. In the setting the location can be managed. In the sign out functionality the user can, not only leave the application, but also save all its interactions when doing so.

As mentioned before, most of the application functionalities are located on the side-bar. Despite the many options a user has, mainly four features can be identified where the user has access to:

1. For a given position, find the closest mushroom picking place for a given species. This is implemented **(Figure 2)** by using user location and calling a function (i.e. *sd_get_closest_mushroom_finding_with_specific_typr_from_lat_lon ()*) that sends a query to the database function that calculates the closest distance from this location to the closest desired mushroom. The user location can be determined automatically by using geolocation or user determined point. The database function calculates the distance for each mushroom and returns the closest one with the desired mushroom specie.
2. For a given position, find the closest nature area with a reported mushroom finding of a specific type and the shortest path within the nature area to the mushroom. The implementation **(Figure 3)** follows the same principle as with the feature one, but it also draws the shortest route from the user location to the closest desired mushroom. The routing is managed by sending the both coordinates to Mapbox routing API, which returns the shortest route as a WMS. For routing feature three different routing options have enabled for

the user: by car, cycling or driving. These choices use different road categories for calculating the shortest path using Mapbox routing API.

3. The third feature of the app **(Figure 4)** allow users to manage their mushroom findings by creating, reading, updating and deleting them (i.e. CRUD functionality). When the *Mushroom Findings tool* is initialized the *getAllUserFindings()* function retrieves all of the user inserted findings to the map if there exist findings, otherwise the map extent is set to the whole Sweden. Then the user can access the *Add New Findings tab* to insert findings. If the user wants to modify or delete the existing findings user can click existing findings and determine whether he/she wants to update or delete these findings.
4. In the Space-time analysis section a user can draw polygons over areas of interest in order to retrieve statistics for that area or find out which is the most popular mushroom finding. It is also possible to make heatmaps for the mushroom findings.

2.2 Backend & Frontend

The application structure (Figure 1) shows the backend of the application. Backend is divided into following parts:

- The web server (Node.js + Express).
- The data store (PostgreSQL + PostGIS). This is where data tables are stored. Personal user data and the mushroom data that is the service that we offer to the users. Here are also the query functions stored as well.
- The map server (Geoserver) that is used for publishing spatial data via WFS.
- External “backend” (Mapbox) for providing routing and tile layers via WMS.

For making user interactions as easy as possible the frontend was implemented as clean as possible. OpenLayers 3 was chosen for the main mapping library for its stability, suitability and extensibility over leaflet (or other mapping solutions) for a GIS web application. Other used frontend technologies besides OpenLayers 3 were:

- JavaScript libraries
  - jQuery, Chart.js, Slick-carousel.js, Mapseed-sidebar.js, MapBox.js (routing)
- Visualization techniques and libraries
  - Basic CSS, Bootstrap + Awesome fonts, Rainbowvis.js

2.3 Data

The data used in the application can be broadly grouped into three kinds: Mapbox styled tilelayer, PostgreSQL tables and Mapbox routing API. Mapbox tilelayer is used as a background map that consists of mapbox street map, terrain, nature areas and walkable roads inside nature areas. This is used in the applications as a WMS layer. PostgreSQL tables consists of user information to check login information, mushroom data that is used for checking mushroom species, user inserted space-time polygons and mushroom findings. Also, for checking that user inserts the mushroom findings only into nature areas, nature areas itself are needed. The list below showcases the used datasets in the application:

- Nature areas consisting of nature reserves, national parks and protected areas. Two-dimensional polygon layer.
- Mushroom species and observations. Data from both sources is combined as an one point layer.
- Space-time polygons
- Mapbox road network served via API
- User account information

2.4 Deployment

The Mushrooming deployment process consists of setting up an account with a VPS Linux server, installing Geoserver at port 8080 and Node.js + Express.js at port 3000. Nginx was installed to act as a (reverse) proxy to forward all different services to port 443 which is the standard HTTPS port. Node.js is forwarded to the root URL and Geoserver to /geoserver. Geolocation in Chrome requires HTTPS. In order to deploy a HTTPS web server a certificate is required. The certificate was obtained from https://letsencrypt.org/, a free certificate authority.

The application is accessed through the URL: www.stiin.se. Therefore, no installation or any other procedures are required. Application can be both accessed from PC’s and mobile devices, in both cases through web browsers. After user creation and login, the user comes to the main view which is clean and simple, providing just the essential elements for the pleasant performing of the activity for which is designed.

2.5 Experimentation & Tests

Testing of the application was done by running the application with many of the popular desktop browsers such as Chrome, Firefox and Opera. Also, in mobile the mobile browsers Chrome and Safari were tested. In the testing process, the purpose was try to find as many bugs as possible by finding every combination of the possible user interactions. To ease this process a series of diagrams were created to comprehend these situations. By creating these diagrams (i.e. Figures 2, 3, 4 and 5) it was possible to easily find the "wormholes" of the application where it might crash or the functionalities are not working as intended.

One of the biggest troubles during the testing phase was to figure out the geolocation functionality. This is because, Google has made geolocation information as secured personal information, meaning that this information should only be distributed by using secured domains. Therefore, using geolocation via Chrome proven to be difficult to implement but in the end we got it working. However, the geolocation is working in all of the mentioned browsers except for Opera. Geolocation functionality was also tested by walking outside and comparing the real position to the one received by our application and the accuracy was enough for our purposes.

Other noticeable remarks that occurred during the testing phase were that browser renders the text (i.e. css) a bit differently. For example if popups of mushroom findings are opened in Firefox the content of it doesn't look as good as in Chrome. Another, noticeable thing is the fitting of the display in mobile devices. This aspect was neglected in most of the application functionalities, since for example the OpenLayers 3 scales its content automatically. In addition, the sidebar plugin has support for mobile users by fitting the sidebar to fit the whole page in mobile phones. Also, it supports multiple screen sizes and fits it accordingly. The display of the sidebar was made so the sidebar opening and closing so that it doesn't interfere the user in mobile nor in desktop.

3. Demonstration

The following subchapters demonstrates the different parts of the application. These subchapter divide the application to the different pages where the user can access into. Also, the functionalities for each pages are explained and illustrated.

3.1 Login / Sign up

The welcoming login or sign up pages (**Figure 6**) were designed to provide a fresh and simple layout somewhat related with the activity the application is designed for. The login/signup functionality uses Passport, an authentication middleware for Node.js (Passport, 2016).

The user information is stored in the database. The password is, for now, not encrypted. It is stored in plain text. If the user credentials are valid, a session is created on the server. The session is stored in a cookie in the web browser so that the login is remembered. The session contains user id, username and email. The user id is used in all database requests when the user id is relevant for the query. For example it is used so that every user can make their own mushroom findings and only update and/or delete his/her own findings. It is also used to retrieve the space-time polygons made in a previous session and store the ones made in the current session.

3.2 Map

The main view (**Figure 7**) is the recipient of most functionalities and events the application has to offer. Hence, most activities are done in the map view or the sidebar, parts in which the main view is divided. The map view serves both as background and as the base for the users interactions. As mushroom activities have a spatial component, is the map what links the spatial location of the users and the objects they aim to access or find. It is also the map view the base for the routing functions, the finding of mushrooms and the spatial analyses.

Although the map view is the base in which all functionalities are based, is the sidebar (**Figure 7**) the recipient of most of functionalities (**Figure 8**). Most event handlings and operations on the side bar are through buttons and dialog boxes, that provides a simple operation layout that even the most inexperienced user can understand. This one of the most important goals of the UI design (**Figure 5**) and behind the whole application enterprise, to provide GIS functionality with a simple design.

Geolocation and Navigation

The user were tracked using Openlayers tracking functionality which is based on the geolocation services provided by web browsers. The locations are then used by the other parts of the app as start points in navigation and to find the closest mushrooms. For users that are disinclined to let their browsers reveal their location can manually set positions instead.

The navigation feature uses Mapbox routing, which provides an API where it is possible to send requests for navigation routes between two coordinates. Mushrooming uses this API to request navigation routes together with the metadata of the route, such as distance, duration, etc.

3.3 Handle mushrooms

In the handle mushrooms section user is admitted for creating, reading, updating and deleting (i.e. CRUD functionalities) mushroom findings. These findings are then user specific and only the user itself (i.e. the one that is logged in) can CRUD management to the findings. When the user goes to the handle mushroom section via clicking a link in the main application the view is set to the according: to the extent of Sweden if the user hasn't set any mushroom findings or to the extent of the inserted findings. (**Figure 9**)

Insert findings

User is able to insert own findings in the application by selecting the second tab inside the sidebar and pressing the "Insert new finding". After clicking the button, user has to zoom as close to them that the nature areas are highlighted. Then the user can click inside the greenery area. This map click will open a sidebar that has fillable form the of the mushroom finding. If user wants to abort the process there is a button for it. (**Figure 10**)

The user input form has many mandatory fields that has to be filled. These fields are shown with a marker (*) besides them. In addition, other restrictions have been added to check the user inserted information. These form restrictions are: mushroom specie cannot be capitalized, mushroom specie has to be a known specie (database check), quantity and precision has to be integers and the date has to be in a correct form. If the user input doesn't match these constraints an error message is shown to the user.

Update and delete findings

Another possibility for the user for handling own mushroom findings is updating and deleting. One restriction related to this is that the user is not able to do these operations without refreshing the page. This is because, the id for each finding is generated inside the database and therefore these has to be retrieved first from the database. However, this is solved by stating an error message if the user doing otherwise.

If user desires to modify existing own findings user has to click one of the finding and the information of it will be shown in the sidebar. The sidebar shows all of the information that has been inserted by the user. Also, the user has possibility to update or delete this findings. By clicking update the update form is shown where the old values are also present. This form also has the same form checking as the insert except the mandatory fields. User can update any combination of fields as desires within the form constraints and press update. Also, the user has an possibility to abort this process. In the delete process the user is asked whether he or she is certain of the deletion of the finding, since this is not possible to redo after applying the delete for the finding. After clicking delete and accepting the procedure the highlighted feature is removed and a label is inserted under the default mushroom icon. After refreshing the page the point is removed from the map. (**Figure 11**)

3.4 Space-time analysis

In the Space-time analysis section a user can draw polygons over areas of interest in order to retrieve statistics for that area or find out which is the most popular mushroom finding. It is also possible to make heatmaps for the mushroom findings. See attached spaceTime.avi screen recording for demonstration of the space-time functionalities.

Space-time polygons

A polygon is stored in the database once the polygon is closed. The polygon geometry can be updated and the database is automatically synchronized. The polygon may be deleted by choosing "Delete polygon" after which the polygon is deleted from the database. If the user has stored any polygons in a previous session they are drawn on start in the new session. All polygons that are drawn on the map will thus be rendered in later sessions. By clicking on a polygon a popup is opened with the alternatives "Most popular" and "Analyse area". The "Most popular" alternative highlights the mushroom finding that is the most popular finding place among the findings within the polygon. Since mushroom findings are not a finite set of locations where findings can be made, it is a bit forced to implement the "most popular finding". However, in order to still implement the most popular mushroom finding functionality, the number of logged mushroom findings was used as criteria. In reality it is not probable users would log the same coordinate for the same mushroom finding place. However, users have gone on excursions and afterwards logged all their findings at a certain coordinate. Therefore, we could still implement the most popular functionality (though in a constrained way).

The "Analyse area" option shows a cumulative findings chart with data from the date of the first made mushroom finding until today's date. It also makes a pie chart displaying the distribution of mushroom species found in the area.

Heatmaps

It is possible to make heatmaps of the mushroom findings. Two kinds of heatmaps are available; static and animated. The heatmaps use the OpenLayers 3's heatmap layer (ol.layer.Heatmap), "a layer for rendering vector data as a heatmap"– OpenLayers 3 API documentation (OpenLayers 3, 2016). The mushroom findings are used as input vector data. No weights or attributes are set. The static heatmap shows all the mushroom findings at once.

Two animated heatmaps are available; hotspot and cumulative heatmap. The animated heatmaps start at the date of the first mushroom finding and stops at the date of the last finding. The hotspot heatmap is a simulation of a hotspot analysis. For each mushroom finding date it renders all the features of the last 20 days that has features, on the map. The cumulative heatmap renders the mushroom findings cumulatively on the map. Thus, it starts with the first mushroom finding and as time goes the heatmap grows according to when and where mushroom findings were made.

4. Conclusions

It can be seen that even though we've got a fully functioning application, much work is to be done. Despite the apparent simplicity of design it takes a lot of time to fine tune functionalities, especially. when many functions are related to each other. Time is been a major constraint to fulfill our group goals with the application, but in general terms, the project guidelines were all completed.

The project was a good learning experiment, since it involved lot of the basic functionalities in Web GIS that can be further enhanced. Not only functionalities related with geolocation -finding mushrooms and finding path to a certain mushroom type- and data management -handling your findings- were implemented, but some statistics and spatial analyses were drawn from the information.

As a result, the application is fully functional and already published with the URL: http://www.stiin.se for users to acess it. Still, as with most applications and web pages much work is still to be done regarding further development of the final product. Hence, some limitations have to be dealt with and much to be discussed about it between the creation team.

4.1 Limitations and Improvements

There are some limitations encountered during the creation process and much room for improvement. The data set is strictly confined to the extent of the protected nature areas, assuming the user is only interested in mushroom observations made within these areas. Furthermore, at the present the saving capabilities of the application are reduced. In addition, the functionality in Opera (web browser) is now currently available, which limits the accessibility range of the application. In addition, some improvement is imperative in the short term time span:

- Improve user information security, encrypt password.
- Possibility to suggest updates and deletes
- User could decide whether to share or not to share findings
- Social media sharing inside the application
- Improved spatial analyses (heatmap + other analyses)
- Improved routing (e.g pgRouting with optimized algorithms)
- Design
    - Every functionality including mushroom handling and space-time analysis should operate on the same page
    - Better aesthetics

4.2 Reflections

The tests have shown some inconsistencies to dealt with for the app to offer the best service possible. Hence, much can be discussed on how to continue developing the application. For instance, the colors in the heatmap were not consistent between the browsers. However, this is not the only inconsistency found during experimentation.

Furthermore, various browsers will execute JavaScript functions in their own way which sometimes are different from other browsers. One such example is the tracking of the user's position. The desktop versions of Firefox, Edge and Chrome return locations continuously. For the desktop version of Opera it was not possible to get any location. The Android versions of the browsers behaved differently compared with the desktop versions as well. We solved this by writing code that would work in all of the discovered browsers mentioned above, except desktop Opera.

Moreover, for the navigation we tried two different solutions. First pgrouting, a problem was that we did not have enough data of the Swedish roads to create routes between most points. Hence, we decided to download data from OSM and import into our pgrouting database. Unfortunately, the program osm2pgrouting crashed several times during the import so we could not use pgrouting, the reason was that the osm2pgrouting conversion didn't succeed. Instead we used Mapbox which has an easy-to-use API for requesting navigation routes between two points and have support for the travel modes driving, walking and cycling. It was enough for us to be able to solve the navigation problems for the application.

4.3 Teamwork

The following list consists of the main responsibilities of each person involved in the project:

Hitomi:

- Geolocation
- Navigation

Kristin:

- Deployment
- User authentication
- Database
- Space-time analysis

Petteri:

- CRUD (i.e. functionalities related to the handleMushrooms)
- Frontend
- Managing functionality design and implementing functionalities (e.g. map click handling, sidebar content info, data handling)

Raul:

- Frontend: UI and styling
- Application testing
- Bug reporting and documentation
- Main lead on report writing

References

Beal, V. (2016). *Web server*. (Q. Inc., Producer, & Webopedia) Retrieved 10 18, 2016, from http://www.webopedia.com/TERM/W/Web_server.html

Bearnes, B. (2016, 05 12). *How To Set Up a Node.js Application for Production on Ubuntu 16.04*. (D. Inc., Editor) Retrieved 10 18, 2016, from DigitalOcean: https://www.digitalocean.com/community/tutorials/how-to-set-up-a-node-js-application-for-production-on-ubuntu-16-04

Esri. (2016). *ArcGIS forServer*. (I. Environmental Systems Research Institute, Editor) Retrieved 10 18, 2016, from http://server.arcgis.com/es/server/10.3/publish-services/linux/what-is-a-map-service.htm

Geoserver. (2014). *What is Geoserver?* Retrieved 10 18, 2016, from Geoserver: http://geoserver.org/about/

OpenLayers 3, 2016. *ol.layer.Heatmap.* Retrieved 30 10, 2016, from http://openlayers.org/en/latest/apidoc/ol.layer.Heatmap.html.

Passport, 2016. *Passport*. Retrieved 30 10, 2016, from http://passportjs.org/

pgRouting Community. (2016). *pgRouting Project*. (p. Community, Editor) Retrieved 10 18, 2016, from http://pgrouting.org/

PostGIS. (2016). *About PostGIS*. Retrieved 10 18, 2016, from http://www.postgis.net/

PostgreSQL. (2016). *What is PostgreSQL?* Retrieved 10 18, 2016, from http://www.postgresqltutorial.com/what-is-postgresql/

Rouse, M. (2000). *Back-end*. (TechTarget, Editor) Retrieved 10 18, 2016, from Search Data Center: http://searchdatacenter.techtarget.com/definition/back-end

Sundfors, M. (2009, 01 31). *Naturplats*. Retrieved 10 17, 2016, from http://www.naturplats.com/svamp/plockasvamp/nyborjarsvampar

Wikipedia. (2015, 07 28). *MapServer*. (I. Wikimedia Foundation, Editor) Retrieved 10 18, 2016, from https://en.wikipedia.org/wiki/MapServer

Appendix

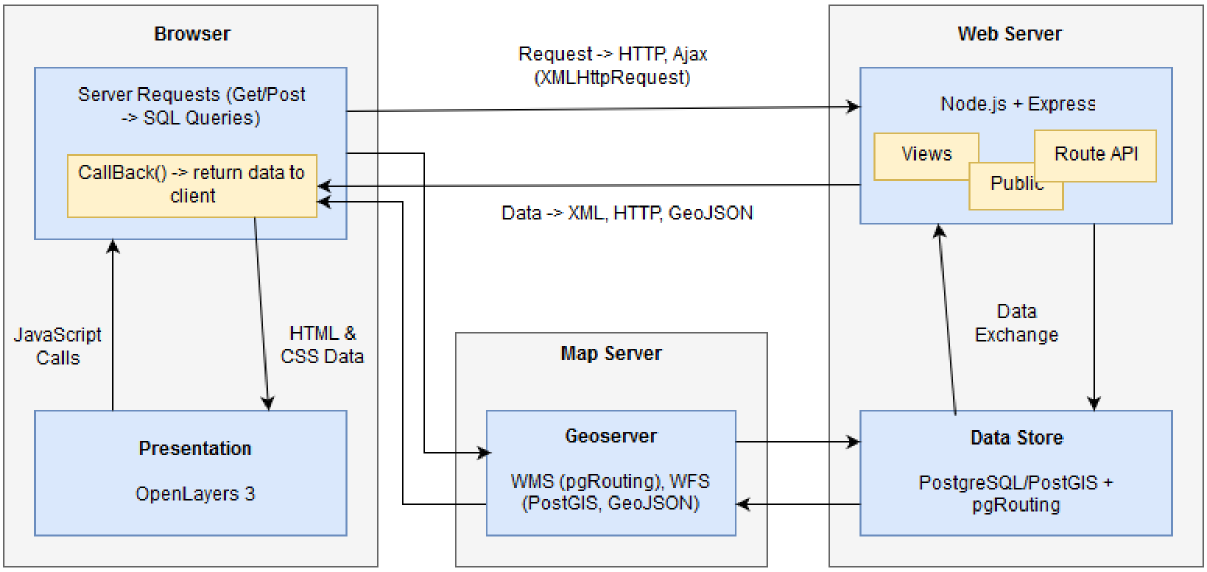

Browser
Server Requests (Get/Post -> SQL Queries)
CallBack() -> return data to client
JavaScript Calls
HTML & CSS Data
Presentation
OpenLayers 3
Request -> HTTP, Ajax (XMLHttpRequest)
Data -> XML, HTTP, GeoJSON
Web Server
Node.js + Express
Views
Public
Route API
Data Exchange
Map Server
Geoserver
WMS (pgRouting), WFS (PostGIS, GeoJSON)
Data Store
PostgreSQL/PostGIS + pgRouting

Figure 1: Web application architecture.

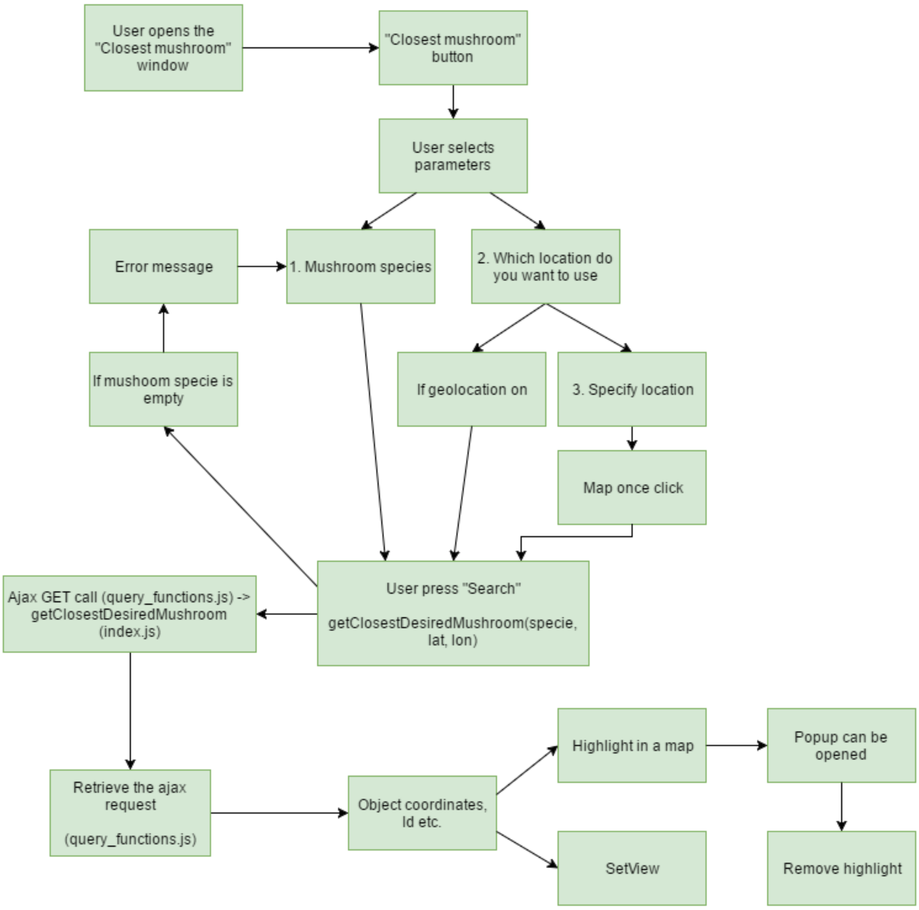

User opens the "Closest mushroom" window
"Closest mushroom" button
User selects parameters
Error message
1. Mushroom species
2. Which location do you want to use
If mushoom specie is empty
If geolocation on
3. Specify location
Map once click
User press "Search"
getClosestDesiredMushroom(specie, lat, lon)
Ajax GET call (query_functions.js) -> getClosestDesiredMushroom (index.js)
Retrieve the ajax request
(query_functions.js)
Object coordinates, Id etc.
Highlight in a map
Popup can be opened
SetView
Remove highlight

Figure 2: Find the closest mushroom of a specific type.

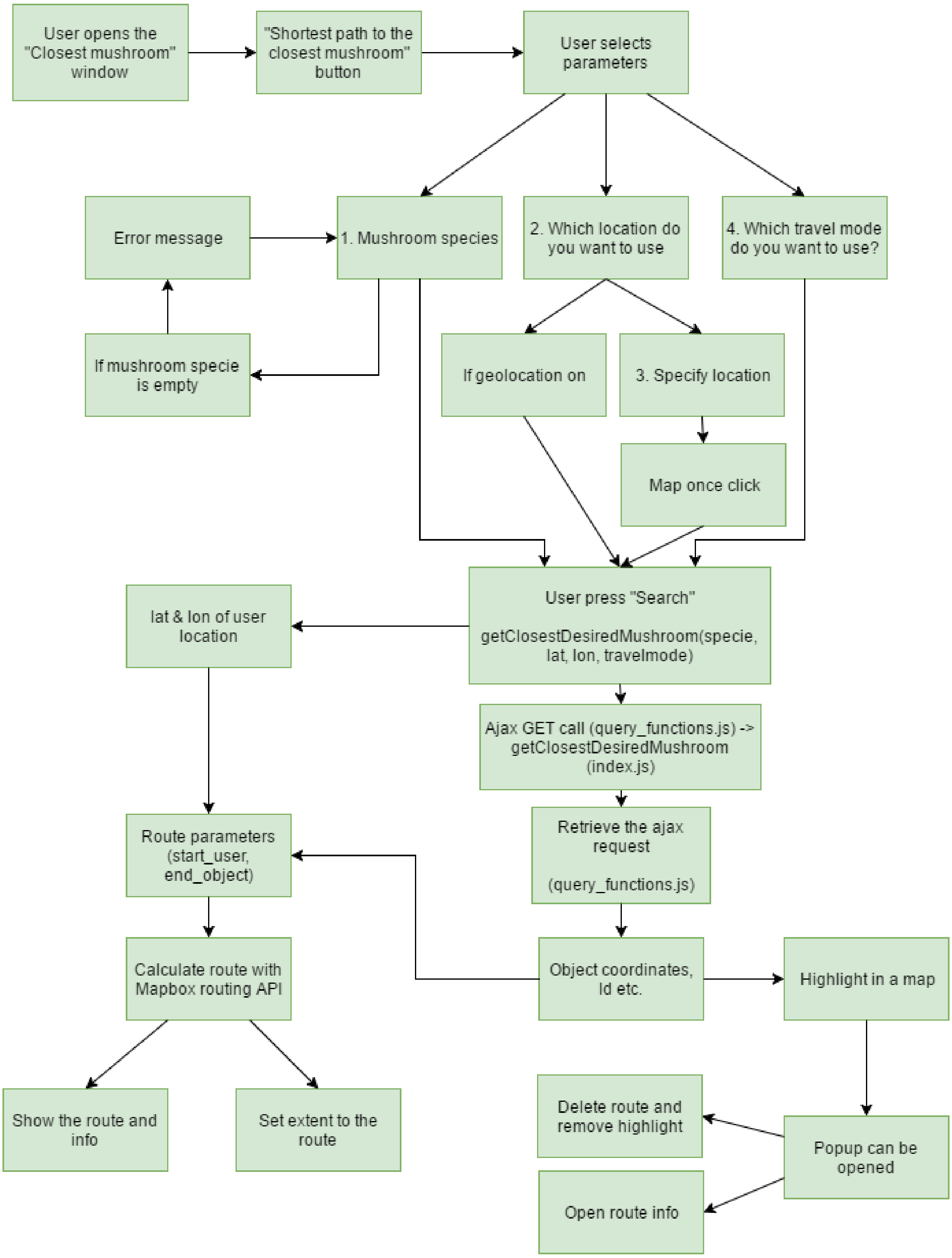

User opens the "Closest mushroom" window
"Shortest path to the closest mushroom" button
User selects parameters
Error message
1. Mushroom species
2. Which location do you want to use
4. Which travel mode do you want to use?
If mushroom specie is empty
If geolocation on
3. Specify location
Map once click
lat & lon of user location
User press "Search"
getClosestDesiredMushroom(specie, lat, lon, travelmode)
Ajax GET call (query_functions.js) -> getClosestDesiredMushroom (index.js)
Route parameters (start_user, end_object)
Retrieve the ajax request
(query_functions.js)
Calculate route with Mapbox routing API
Object coordinates, Id etc.
Highlight in a map
Show the route and info
Set extent to the route
Delete route and remove highlight
Popup can be opened
Open route info

Figure 3: Find the closest mushroom of a specific type and the shortest path there.

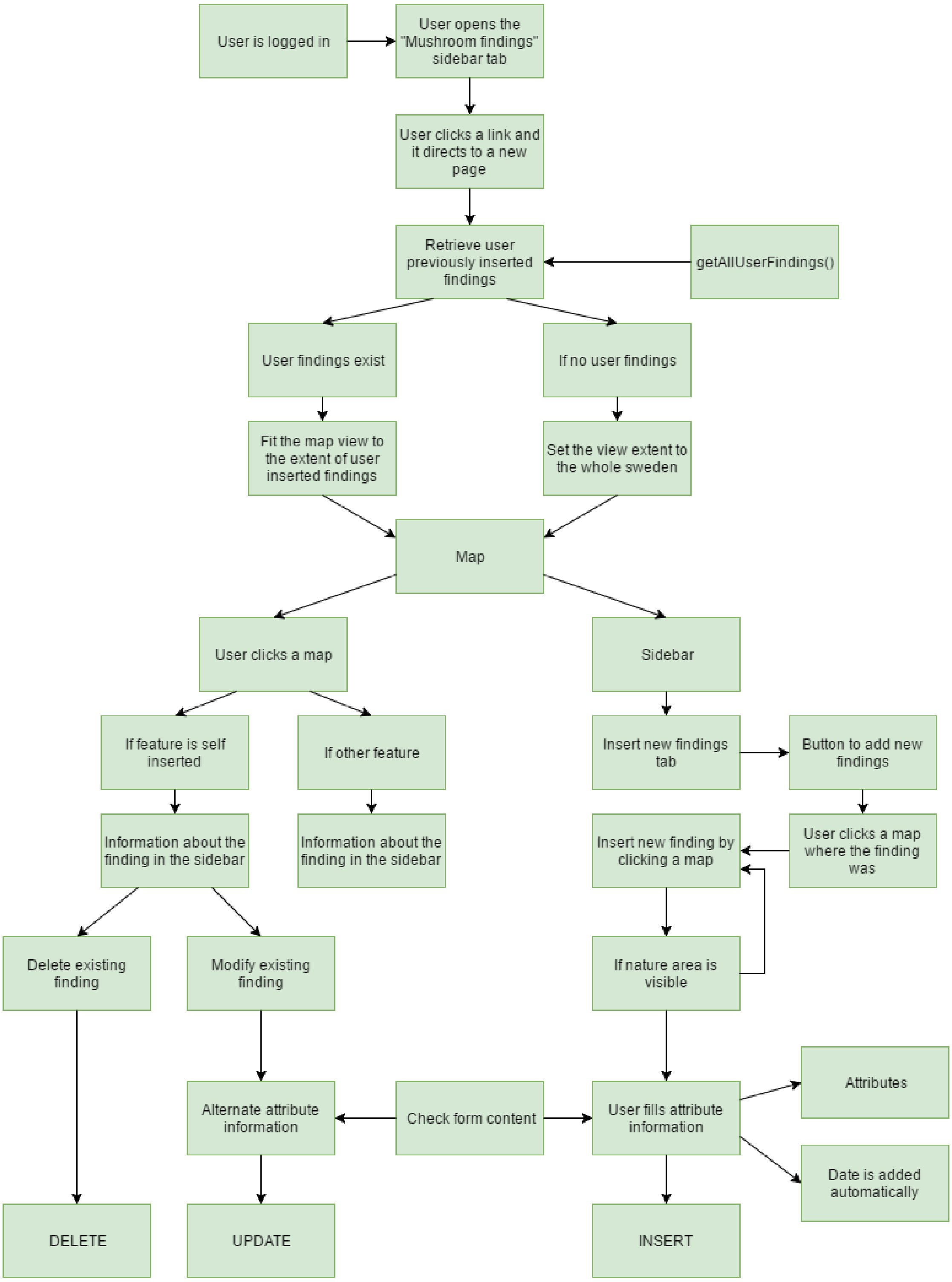

User is logged in
User opens the "Mushroom findings" sidebar tab
User clicks a link and it directs to a new page
Retrieve user previously inserted findings
getAllUserFindings()
User findings exist
If no user findings
Fit the map view to the extent of user inserted findings
Set the view extent to the whole sweden
Map
User clicks a map
Sidebar
If feature is self inserted
If other feature
Insert new findings tab
Button to add new findings
Information about the finding in the sidebar
Information about the finding in the sidebar
Insert new finding by clicking a map
User clicks a map where the finding was
Delete existing finding
Modify existing finding
If nature area is visible
Attributes
Alternate attribute information
Check form content
User fills attribute information
Date is added automatically
DELETE
UPDATE
INSERT

Figure 4: Add new findings and modify existing ones.

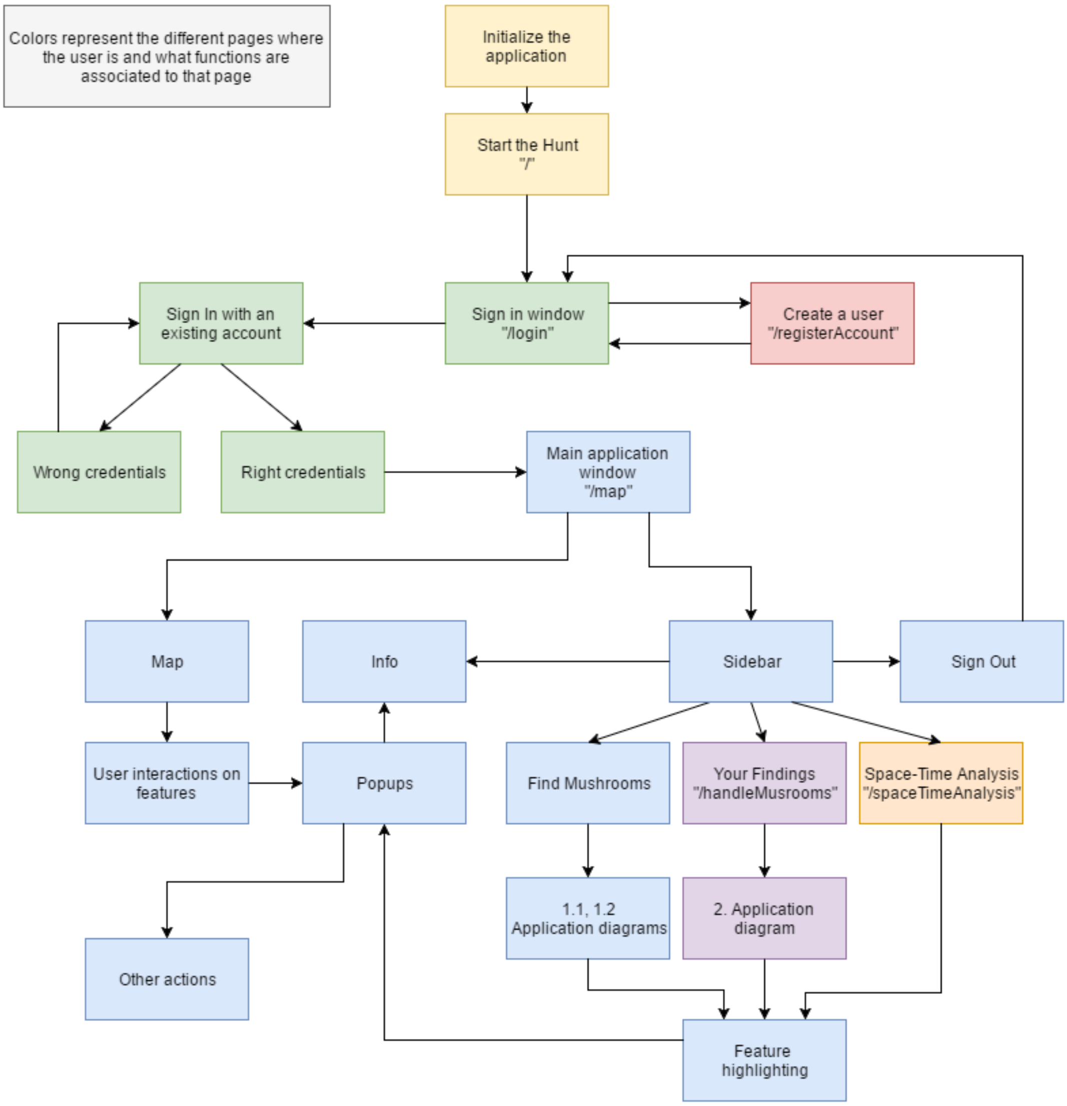
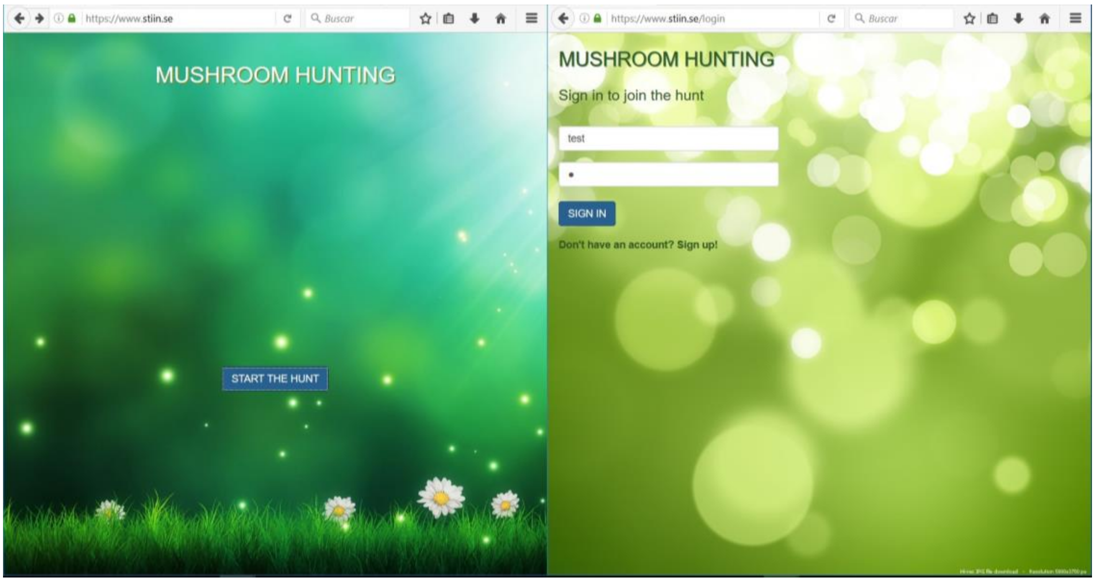

Colors represent the different pages where the user is and what functions are associated to that page
Initialize the application
Start the Hunt "/"
Sign in window "/login"
Create a user "/registerAccount"
Sign In with an existing account
Wrong credentials
Right credentials
Main application window "/map"
Map
Info
Sidebar
Sign Out
User interactions on features
Popups
Find Mushrooms
Your Findings "/handleMusrooms"
Space-Time Analysis "/spaceTimeAnalysis"
1.1, 1.2 Application diagrams
2. Application diagram
Other actions
Feature highlighting

Figure 5: User interface functionality.



MUSHROOM HUNTING

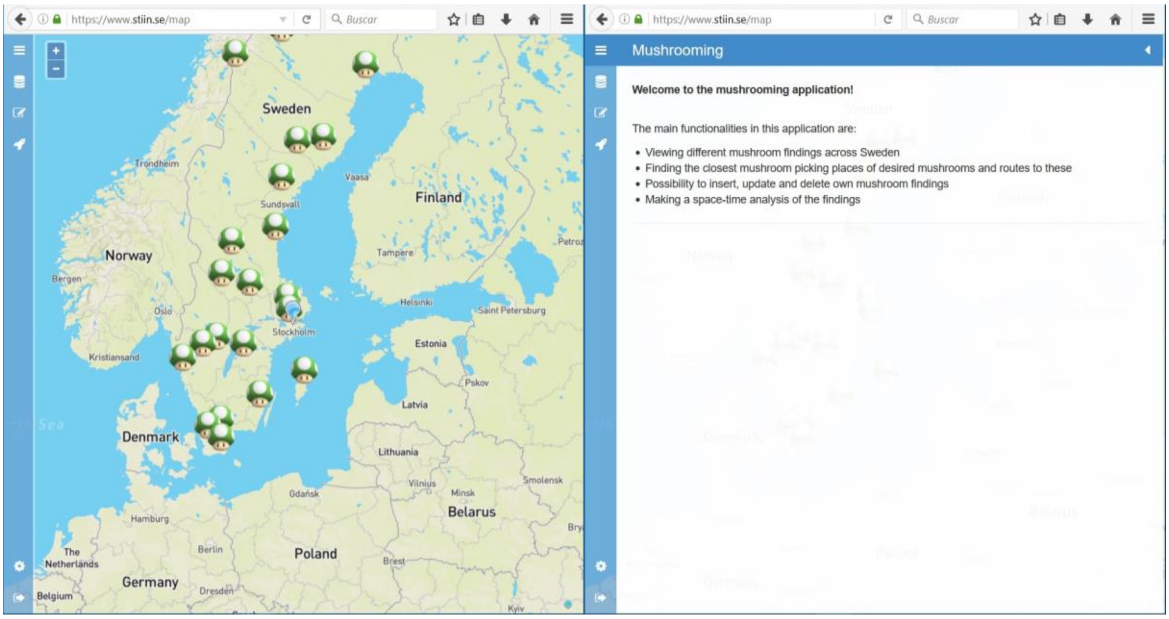
MUSHROOM HUNTING

Sign in to join the hunt

test

Don't have an account? Sign up!

Figure 6: Login/create user module.



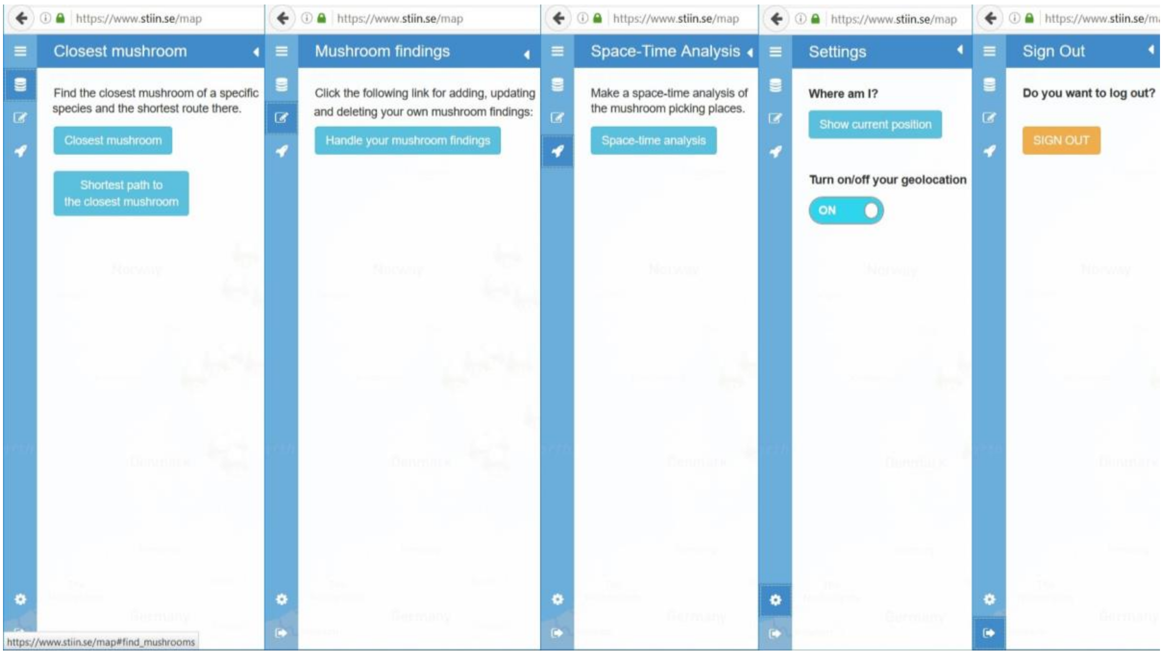
Mushrooming

Welcome to the mushrooming application!

The main functionalities in this application are:

- Viewing different mushroom findings across Sweden
- Finding the closest mushroom picking places of desired mushrooms and routes to these
- Possibility to insert, update and delete own mushroom findings
- Making a space-time analysis of the findings

Figure 7: Main view.



https://www.stiin.se/map
https://www.stiin.se/map#find_mushrooms

Closest mushroom

Find the closest mushroom of a specific species and the shortest route there.

Closest mushroom
Shortest path to the closest mushroom

Mushroom findings

Click the following link for adding, updating and deleting your own mushroom findings:

Handle your mushroom findings

Space-Time Analysis

Make a space-time analysis of the mushroom picking places.

Settings

Where am I?

Turn on/off your geolocation

Sign Out

Do you want to log out?

Figure 8: Sidebar functionalities.

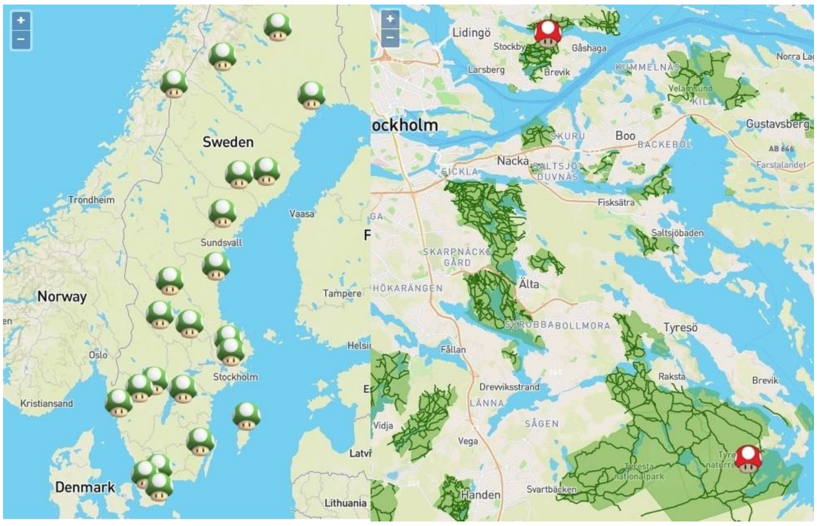
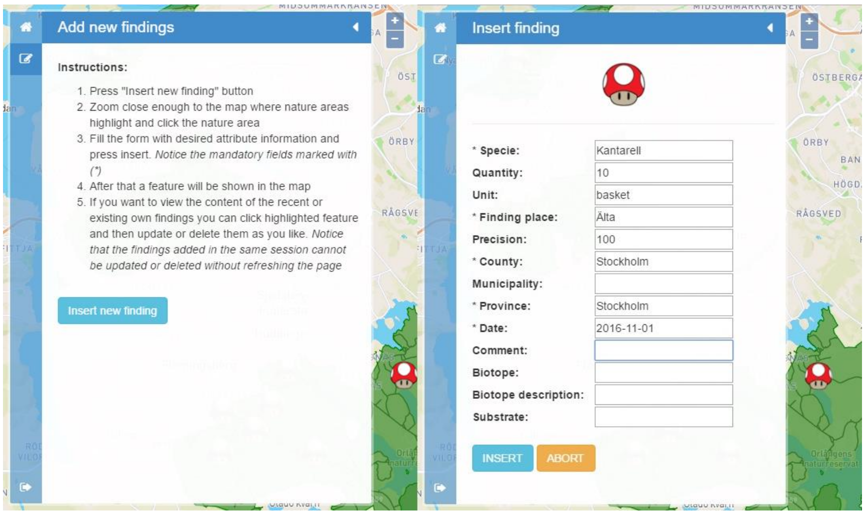


Figure 9: Handle mushrooms view when user has not inserted findings vs. inserted findings.


Add new findings

Instructions:

1. Press "Insert new finding" button
2. Zoom close enough to the map where nature areas highlight and click the nature area
3. Fill the form with desired attribute information and press insert. *Notice the mandatory fields marked with (*)*
4. After that a feature will be shown in the map
5. If you want to view the content of the recent or existing own findings you can click highlighted feature and then update or delete them as you like. *Notice that the findings added in the same session cannot be updated or deleted without refreshing the page*

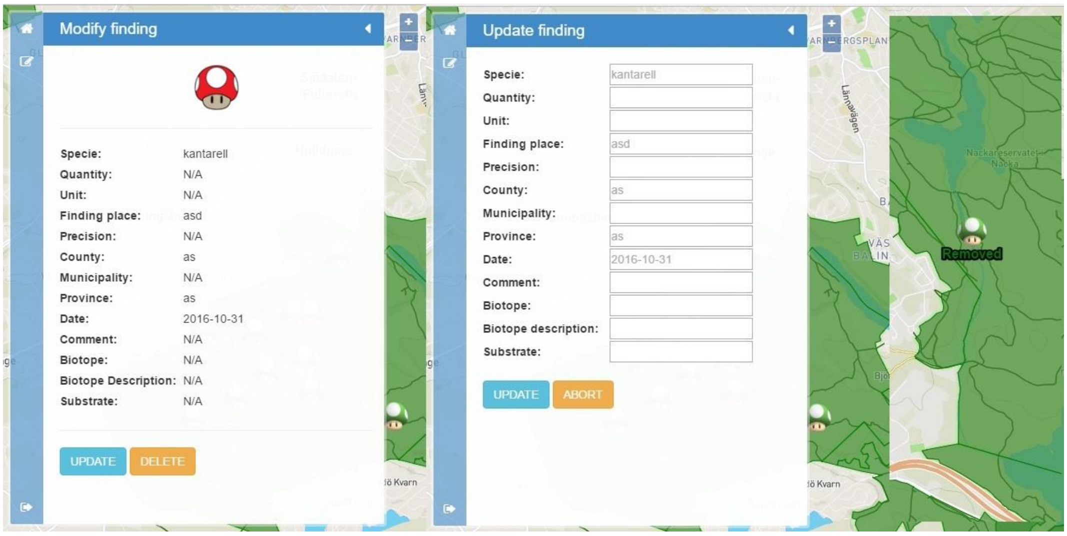
Insert new finding

Insert finding

* Specie:	Kantarell
Quantity:	10
Unit:	basket
* Finding place:	Älta
Precision:	100
* County:	Stockholm
Municipality:	
* Province:	Stockholm
* Date:	2016-11-01
Comment:	
Biotope:	
Biotope description:	
Substrate:	

INSERT ABORT

Figure 10: Inserting a mushroom finding.


Modify finding

Specie:	kantarell
Quantity:	N/A
Unit:	N/A
Finding place:	asd
Precision:	N/A
County:	as
Municipality:	N/A
Province:	as
Date:	2016-10-31
Comment:	N/A
Biotope:	N/A
Biotope Description:	N/A
Substrate:	N/A

UPDATE DELETE

Update finding

Specie:	kantarell
Quantity:	
Unit:	
Finding place:	asd
Precision:	
County:	as
Municipality:	
Province:	as
Date:	2016-10-31
Comment:	
Biotope:	
Biotope description:	
Substrate:	

UPDATE ABORT

Figure 11: Updating or deleting existing mushroom findings.